2009 Agile Conference

XP Customer Practices: A Grounded Theory

Angela Martin
The University of Waikato, Hamilton, New Zealand
angela@cs.waikato.ac.nz

Robert Biddle
Carleton University, Ottawa, Canada
robert_biddle@carleton.ca

James Noble
Victoria University of Wellington, New Zealand
kjx@mcs.vuw.ac.nz

Abstract

The Customer is a critical role in XP, but almost all XP practices are presented for developers by developers. While XP calls for Real Customer Involvement, it does not explain what XP Customers should do, nor how they should do it. Using Grounded Theory, we discovered eight customer practices used by successful XP teams: Customer Boot Camp, Customer's Apprentice, Customer Pairing, and Programmer's Holiday support the well-being and effectiveness of customers; Programmer On-site and Roadshows support team and organization interactions; and Big Picture Up Front and Re-calibration support Customers steering the whole project. By adopting these processes, XP Customers and teams can work faster and more sustainably.

1. Introduction

> *One of the things that is unsaid in the XP literature is how to be a customer (analyst)*
> *—Martin Fowler* [1]

In the second edition of his XP book, Beck [2] introduced three overarching practices: Real Customer Involvement, Whole Team, and Energized Work. Real Customer Involvement emphasizes the direct involvement of end-users and other business stakeholders on the project. Whole Team refers to the practice of including all of the skills and perspectives on the team necessary for it to succeed. Importantly, Whole Team emphasizes the importance of the sense of team, all team members sharing a sense of purpose and supporting each other. Energized Work emphasizes working "only as many hours as you can be productive and only as many as you can sustain". We wanted to know how teams implement these practices, and we were especially interested in the Customer role. We have conducted extensive qualitative studies of XP teams, and identified eight new customer-focused practices that emerge, all contributing to the three more abstract higher-level practices suggested by Beck (see figure 1).

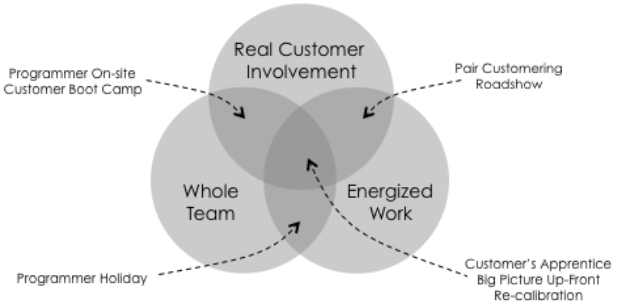

Figure 1: XP Customer Practices

In this paper we will discuss each of the identified practices individually. In our studies we also observed wide-spread use of the already well-known customer practices including Planning Game, Short Releases, Stories and Tests [2; 3]: we do not discuss these or any of the standard business practices such as market research, business process modeling, and so forth.

2. Research Method

Information Systems Development (ISD) methodology researchers [4; 5] have expressed a growing concern that existing ISD methods do not meet the needs of today's business and software development environments. Studies [5] in this area have begun to explore practices in natural settings in order to begin to address these issues. Given this trend, we have used grounded theory [6] to explore our research questions within their natural setting, software projects. We used semi-structured in-depth one-on-one interviews as well as project team observations to collect the data for this paper. Eleven XP projects are explored; we interviewed a total of 66 people across the eleven projects. Our interviews covered the spectrum of core XP roles including the big boss, customer, programmer, coach and tester. All interviews were taped and later transcribed in detail. The interviewees

978-0-7695-3768-9/09 $25.00 © 2009 IEEE
DOI 10.1109/AGILE.2009.68

were asked to validate both the transcriptions of the interview and the interpreted findings. The project observations were used to support both the interview process and the resulting findings. In the sections that follow, we identify the practices as they emerged from our analysis. We use a number of quotes from the interviews to illustrate our findings; names have been avoided or invented to preserve anonymity. The teams we studied were working on a variety of projects:

Company	Location	Domain	Size*	Team
KiwiCorp	Australasia	Telecoms	M	11
RavenCorp	USA	Research	S	20
EagleCorp	USA	Software Tools	S	16
FalconCorp	USA	Retail Products	S	60
SwiftCorp	USA	Retail	XL	20
HawkCorp	USA	Knowledge	S	5
TernCorp	Europe	Telecoms	M	20
SparrowCorp	UK	Energy	L	15
KiteCorp	Europe	Transport	M	12
RobinCorp	UK	Mobile software	XS	6
OwlCorp	UK	Health	L	20

* Companies were classified by the number of employees: XS < 15, S < 1,000, M < 10,000, L < 100,000 and XL 100,000+

This study was part of a doctoral programme. The resulting PhD thesis [7] provides a full description of our application of Grounded Theory, including details of the selection of interviewees and teams, and a full literature review. Space restrictions did not permit us to include the details in this paper.

3. Customer's Apprentice

Quick Definition: Programmer works on the customer[1] team for an iteration or two so that they can understand the complexity of the customer teams' role.

We observed that people on effective whole teams exhibited empathy [9] and respect for other team members and the roles they played on the team. It emerged that one way for developers to understand the Customer was to "walk a mile in their shoes". We first encountered the dramatic nature of this experience in a story from SwiftCorp. The SwiftCorp team all described a period where the programmers were becoming increasingly frustrated with the customer as the customer was providing them with insufficiently detailed stories (and sometimes simply not enough stories for a complete iteration):

> *"[Programmer] said why don't I ... go and write stories with our customer and that'll help him out ... And if it turns out to be really easy then we'll continue to hit our customer and we'll track spikes through his chest. But that's not what happened. He came back and said oh wow, there's really a kind of a process to writing stories and to mine requirements and go hunt down the people in the business that want something and to get them to explain what it is that they want and so on and so forth ... It humbled them a little bit I think and that made the relationship much more productive ... And it completely turned things around for that group. In that sense the developer had a little bit of insight into the pain in the customer's world."*
>
> *—Programmer Coach, SwiftCorp*

In doing this, the programmer did indeed help the customer move forward, but perhaps more importantly, he gained a deeper understanding and appreciation of the extent of the customer's task, and was able to take that understanding and empathy back into the team. This event was critical in helping the team work together more effectively.

The SparrowCorp team related a similar experience on their project. In this situation the programmer-customer bond was enhanced when the programmers attended the sessions that the customer held with external stakeholders. The programmers undertook some technical installation tasks at each visit, but more importantly they got to see the other 'face' of the customer, the positive way she projected the software and the team, and they got to see the pressure she faced from external stakeholders.

A number of experience reports [10; 11] also describe situations that led to opportunities for the Customer's Apprentice practice to be used. Hodgetts [10] describes a Government Workflow Project where the business experts were not skilled in analyzing the processes and creating a specification for the programmers, and so an "analysis backlog" developed. The team resolved this issue by two of the programmers moving over to assist (and coach) the business experts for a couple of iterations. The benefits were two-fold, the business experts developed new skills, and were able to better meet the needs of the programmers, and the programmers were able to understand and help address the issues that had begun to reduce programmer morale on the team. Takats and Brewer [11] write of their experience at Sapient, while working with the U. S. Office of Naval Research (ONR). On this project, a few of the senior programmers were selected to assist as note-takers (or in some cases facilitators) during initial sessions that occurred on the project. This allowed them to gain a direct insight into the domain, and create relationships

[1] Our paper [8] clarifies the structure of the customer team

with the customer team prior to the development phase of the project starting.

It also emerged that programmers who have played the role of the Customer's Apprentice are more likely to see the team as a whole team. This change helps to move us closer to the vision Beck had of the Whole Team practice [2], creating a stronger and more effective business-technical collaboration. In each case the programmer assisted in reducing the burden of overload from the customer, improving their ability to experience Energized Work.

4. Programmer On-site

Quick Definition: Schedule site visits for programmers so that programmers can understand more about the end-users of the software.

Beck aspires "to reduce wasted effort by putting the people with the needs in direct contact with the people who can fill those needs" [2]. In our study we noticed that programmers were keen to better understand or connect with the direct end-users of the system:

> *"I've always felt bad that we never talk to the customers. Like engineers ... don't actually go out and have meetings with the customers. [EagleCorp] has their Customer Advisory Council ... but we just get features ... we don't see how people would use it."*
> *—Programmer, EagleCorp*

Similar comments were discovered in other cases where we had a product manager or business analyst playing the onsite-customer. In situations like OwlCorp, where we had two end-user representatives available to the team full-time, we did not see comments like these from the programmers. It is not as simple, however, as simply putting end-users and programmers together. At RavenCorp, for example, despite a full-time end-user representative being on-site, there was still a lack of understanding of what the end-users were trying to accomplish:

> *"I guess the other thing that I would change ... [is] the software developers ... don't have as much of an interest in what we're doing on the science side as far as what ... the product we're producing is going to mean to [the domain] in general and if they could ... grasp how useful ... just how cool what we're making is ... then they['d] enjoy working [and] ... be more committed to this company ... not just be someone who, [comes] in to do their 8 hours of work..."*
> *—Scientist (Customer), RavenCorp*

Some recent papers [12; 13] have begun to consider Real Customer Involvement, most turning in some fashion to UCD (User-Centered Design) for inspiration. Beyer, Holtzblatt and Baker [12] provide some insight into involving real end-users in the project. Their recommendation is to use contextual inquiry, observing the end-users in their day-to-day activities, and then providing summarized models from those observations to inform the larger project team. The additional, and in the context of this research important, recommendation is to include programmers as part of the cross-functional contextual inquiry team. Beyer et al. note that the inclusion of programmers on this team does not always occur in practice, but then recommend involving them as early as possible afterward and making the programmers aware of the contextual inquiry findings. Broschinsky and Baker [13] combined the use of contextual inquiry and personas, with one of the models resulting from their use of contextual inquiry being a set of personas. They noticed that the data only resonated with the programmers on their team once they brought actual or real end-users in to meet the programmers. It was at that point that their findings became real and believable for programmers.

Moreover, while we have tended to focus on the impact of these techniques to the programmers in the above paragraph, it is essential to remember that the benefits are always two-fold. Beyer [12] highlights that a number of misconceptions arise with Beck's concept of Real Customer Involvement including that "people cannot articulate their own work practice" and end-users and other stakeholders "are not designers". Programmer On-site is concerned with the whole team understanding the end user and context of use, with the programmers gaining enough information to make helpful suggestions.

5. Programmer Holiday

Quick Definition: An iteration of technical tasks so that the customer can have some time to think-ahead.

We noticed that teams increase the number of programmers assigned to a project slowly at the start of a project. The programmers who join early will often be working on technical tasks, not driven by stories, such as setting up the technical environment. We wondered if there was ever a need to replicate that situation once the programmer team has ramped-up to full capacity? We observed that the XP iteration driven approach is intense both for the business and technical sides of the team:

> *"One of the ways the [developers] deal with [the intenseness of XP] is to [have] the opportunity ... [to] choose during the iteration cycle to step out of the [story-*

driven] development process and to work on something that's more of a sideline, so they'll work on something that's supporting the iteration ... [called a joker card]"
—Customer, EagleCorp

These non-story tasks included repaying technical debt, upgrading software or hardware, developing a tool to support development (e.g. a code generation tool) or conducting research into new technologies. Mackinnon [14] describes in his experience report a similar concept, involving what the Connextra team named "gold cards". They discovered that gold cards helped improve the team morale and reduced the monotony of the iterations for programmers.

The SwiftCorp Coach discussed one team who took this joker or gold card practice to an extreme and had a whole iteration for refactoring:

"[the company] use to have re-factoring iterations, so they would let the team re-factor for a whole iteration ... their rule was that at the end of the iteration all the stories ... and all the unit tests ... run and ... they could do whatever they wanted under the covers. It's a little bit like giving all the developers, I mean pardon me for saying this about the developers 'cos I was one, so I can say it's like here's some lollipops and popcorn, you know, have a good time..."
—Coach, SwiftCorp

So a mechanism to provide a "time-out" for programmers emerged, but how could a time-out of the iteration process be provided for a Customer? The Customer drives the iteration process, so a time-out for the Customer automatically appears to have the result of programmers not having enough stories for an iteration. We faced this situation on an XP project where the project was two and a half months in, one release had been made, and the second was well on the way to completion, but the customer team was not quite sure what the functionality for release three should be. The third release was critical, but they were not sure what stories would be needed to meet its goal; they were not sure how to "break the back" of the problem. The Customer team needed time to think; and the programmers were carrying a lot of technical debt and had not had a break from the story-driven iteration cycle to fix the technical debt and to research a new build and testing tool. Inspiration arose from the findings that had emerged on the research to date, and an agreement was struck. The programmers would spend an entire iteration on technical tasks of their choosing, and the customer team would step away from the iteration process and "break the back" of the third release. Typically, some of the programmers would work as a Customer's Apprentice during this period.

Programmer Holiday directly contributes to Energized Work for the whole team, but most importantly, from the perspective of this research, for the customer. The customer gets a break from supporting the current iteration and is able to focus almost solely on setting the direction for the next stage of the project. It is these aspects that provide the business value for a Programmer Holiday, although gold cards enable refactorings that may also have significant value.

6. Roadshow

Quick Definition: Demonstrate the software to end-users and other stakeholders so that the team can obtain feedback on the direction of the project.

Beck [2; 3] would like Real Customer Involvement on XP projects, and more specifically he would like opportunities for end-users and other stakeholders to provide feedback on the software as it evolves. Software systems often have a large number of end-users and stakeholders. It emerged that the customer team typically included end-user representatives, who work with the team to represent the perspective of different sections of this community. But how did the customer obtain the feedback on the software from the larger communities they represent? Most of the projects we studied used a practice we have called Roadshow to do this.

EagleCorp, a software product development company, described their Roadshows, and the different audiences of their Roadshows, in some detail. We noticed that the intent of the Roadshows varied slightly based on the interests and needs of that audience, as well as what the customer team needed from that audience.

EagleCorp used a Roadshow to reach an internal audience. The internal audience consisted of the sales, marketing, operational support departments and the executive management team. The team used the Roadshows primarily to report progress and gain feedback on the functionality in-development. An additional side effect of these internal Roadshows was that the executive management was assured that the team was making demonstrable progress towards a shippable product.

EagleCorp also used a Roadshow to reach their many external audiences. The product manager described their interaction with the Customer Advisory Group (CAG). This group met on average once a quarter, with two of those meetings typically being multi-day face-to-face meetings, and the other two

meetings typically being a much shorter two hour webinar. The customer describes the importance of two-way communication during these sessions:

> *"The customers [will] present to us how they're using the application at their organisation. So they go through and tell us, or they show us what business problems it solves for them, how they use it, how their groups use it, and then they'll tell us some of the challenges that they have – whether it's business level challenges with adoption inside their organisation … it's very valuable because it helps me understand what some key customers are doing, some of the key issues that they have … and how I should, spend my engineering … dollars actually building the next generation tools."*
>
> *—Customer, EagleCorp*

The key focus of the Roadshow is demonstrating the product and getting feedback on what has been developed since the last meeting, as well as what is on the radar for the next development period. One of the advantages that XP gave them as a software product development company was the ability to demonstrate working software rather than discussing ideas or using 'smoke and mirror' prototypes:

> *"I think it makes a lot of difference that, again, until it's something that they can really look and feel, that they can actually touch … this industry is still, you know, selling on a promise a lot of the time. You know, [clients] over a course of time have just become very disillusioned with selling them promises. So, just being able to show them the application and show them the fact that it is functionally working, it makes a lot of difference …"*
>
> *—Customer, EagleCorp*

Scrum has a process that initially appears to be the same as the Roadshow practice we describe; the Scrum practice is called a *Sprint Review* [15]. Both practices involve a regular meeting with interested project stakeholders to review the functionality developed. One of the biggest differences between these two practices is that Roadshows are tied into the pulse of the organization rather than solely to the pulse of the development team.

7. Customer Pairing

Quick Definition: Two members of the customer team working collaboratively to provide a single-voice to the development team.

We noticed the practice of customer pairing in varying degrees in many of the cases; it particularly stood out in OwlCorp, where two end-users paired on the team almost 100% of the time:

> *"The work was quite evenly distributed. We would share a lot of ideas in conversations that we had, and we would discuss practically everything. We wouldn't make decisions on our own very often. We would always ask the other person and discuss it to make sure everything was covered. I think that helped as well, to make sure what we were asking for was right … to not to have had a second opinion would have been very difficult… I think it would be just too hard to be the only business-person surrounded by 10 or 15 technical people. It is nothing against them, because I really like them all, but you know sometimes, you just need someone [customer pair] to see it from your point of view."*
>
> *—Customer, OwlCorp*

Over the course of our interviews at OwlCorp we discovered that some of the most important aspects of customer pairing are that the pair can:

- Support each other to make tough decisions,
- Bounce concepts and decisions off each other,
- Sanity check their interpretations of meetings

At OwlCorp the pair spent almost 100% of their time as a collaborative pair. In most of the other cases where we observed customer pairing, the pair utilized a divide-and-conquer strategy that allowed them to both work independently as well as collaborate as a pair. An illustrative example of this strategy comes from SparrowCorp. At SparrowCorp one of the business analysts was responsible for the requirements and needs of six of the regions affected by the system and the other business analyst was responsible for a similar number. This division of labor resulted in the business analysts being able to work independently, and develop the strength of relationships required with each region. However, they brought that information back into the pair in order to forge a single-voice for the development team. The additional sounding board effect (or two heads are better than one) that the OwlCorp customer refers to also helps to create less stress for the customer that again helps to facilitate Energized Work.

8. Customer Boot Camp

Quick Definition: A customer-focused training event.

How do the customer team, be they from a traditional business analyst or from a business

background (e.g. an end-user representative), learn how to interact with an XP development team effectively? The business analyst team leader at SwiftCorp noted it as a potential issue:

> *"[XP] is new to everybody ... and because it is not a methodology that the team, the BA team has adopted, they've not been trained on it, so it's an adjustment, so that [is] like one of the biggest challenges. Not knocking the process, again I think the process is excellent, it's just that the company hasn't adopted it in density ... and until they do, you are going to have teams that have never done it, and so they are a little disorientated and apprehensive."*
>
> *—Customer, SwiftCorp*

The OwlCorp coach, aware of this type of issue, suggested the team have a special customer boot camp that trained the customer in the agile process and their role. The training would involve a number of representatives from the team but would be focused on the customer's perspective. The key aims were to help people buy in to the process, and to gain a practical understanding of their role and what they need to do on the project. It emerged that the boot camp did not answer all of their questions, and neither did they retain everything, as they often needed to try to do some things in real-life before all of the concepts embedded:

> *"It wasn't until I started to do it that I started to realise what everything was ... I didn't feel comfortable at the finish of boot camp that I understood all that perfectly. I understood bits and pieces."*
>
> *—Customer, OwlCorp*

However, the customer boot camp provided the customer with a "kick-start", an initial understanding of their role, the process, and some initial ideas of techniques, like story writing, that they would be expected to put into practice during the project.

A number of recent experience reports [16; 17] have also reported the importance of including customer-focused training sessions as part of an agile adoption effort. Ganis et al. [16] write of their use of the *Extreme Construction* game when introducing XP into their environment at IBM. This non-software simulation involves specifying and building a physical model of a product using arts and crafts materials. The team invited their customers to attend their agile training, and as such the customers were quickly exposed to the ideas, principles and practices of XP. The non-software simulation allowed both technical and customer team members to gain an appreciation for all of the XP practices.

Rasmusson [17] writes of his experiences on a number of Thoughtworks agile adoption consulting engagements. He writes of two practices that he uses, one being a four-day boot camp that occurs near the start of an engagement. The four days are broken down into two days that focus on aspects relevant to the whole team, including an introduction to agile, roles and responsibilities, release planning and team practices.

The Customer Boot Camp practice supports the customer to become an effective member of the whole team, as it helps them understand more about their role and responsibilities. Therefore, we believe that this practice helps us to obtain Real Customer Involvement as well as help us move towards a true Whole Team that includes the customer.

9. Big Picture Up-Front

Quick Definition: A short period of envisioning amongst the business stakeholders and project team to set the direction of the project.

In our studies it emerged that typically the customer engaged in some activities prior to the first iteration with the development team. The intent of these activities was to help answer the question "what to build", and to set the direction of the project.

At TernCorp the initial project concept or goal was first seriously considered by the business organization over a year before the full project team started work. The software project began with a 14-day period where the end-user representatives worked with some of the members of the project team to create an initial *big picture* for the project. For this project the big picture consisted of a set of use cases and a release plan. During the project the release plan was on the wall of the project room and was a series of post-it notes on brown paper.

In our SparrowCorp case study it also emerged that significant investment occurred prior to the software portion of the project:

> *"So it's a pretty big system. $12 million budget, 10 man years development .. and they didn't have any agreement with any of the countries that they would implement what was developed. So I said ... let's do this properly and get agreement from every country because there is no point building a solution if nobody is going to use it"*
>
> *—Project Manager, SparrowCorp*

The output from this big picture phase included an understanding of the as-is process, both by the regional business units and the analysts, an agreement for a new

process, a release plan with four release milestones and an associated set of stories.

A number of published papers [11; 18] highlight Big Picture Up-Front activities on agile projects. Fuqua and Hammer [18] explain that one of the key lessons from their project was: "don't try to find all of the stories up-front, and expect to throw many away". Fuqua and Hammer don't suggest the removal of that initial conception phase, but do suggest shortening its duration. Takats and Brewer [11] describe their experiences of developing a big picture for a naval logistics command and control system, using visual models and a series of workshops to bring all of the stakeholders together to "own" the big picture.

Big Picture Up-Front also supports Whole Team, as developers are included in the workshops to help build their domain understanding and improve the estimating process. Finally, Big Picture Up-Front also supports Energized Work for the customer team.

10. Re-Calibration

Quick Definition: Plan to adjust commitments and resources regularly based on what both customers and developers learn during the iterations.

After a few iterations, many teams realize that they are not going to deliver everything that they initially hoped they could during release planning. The velocity data from the first iteration will typically indicate that the release plan is unrealistic, but both customers and developers typically attribute this to the effects of a new process or technology. Their expectation is that they will improve and catch up. After a few iterations, however, the team begins to gradually realize that the plan was overly optimistic. We noticed that the reaction at this point begins to differ slightly between the business and technical sides of the team. The technical side of the team perceives that it was "good" that they had uncovered this situation as early as they had. Their perspective: it allows the business team to make the scope reductions required in order to meet the deadline. The business side of the team, however, perceived the situation as more problematic:

> *"I think we agreed what would be a realistic range and what would be a stretch range. Not really kind of knowing what that would mean in terms of an outcome … we had been told … that we could have all the musts, all the shoulds, and some of the coulds … And then it got to a point where we found out that we couldn't have any of the coulds, and there was some coulds that really should have been shoulds … If we were really truthful, it was a bit of a quick and dirty prioritisation and then we were kind of held to it. I guess we were a bit naïve."*
>
> *—Big Boss, OwlCorp*

The primary consideration for the business team during this period is whether any reduced set of functionality will be sufficient to deliver the necessary business value. Some stakeholders remember this process with a great deal of negative emotion. Interestingly enough, in some cases the customer's sense of "betrayal" appeared to be greater than if the situation occurred on traditional projects. One explanation perhaps is that the customer believed XP was a silver bullet.

Beck [3] discusses the planning strategy of XP, and specifically outlines a *steering phase*. The intent of the steering phase is to update the plan based on what the team learns, including new stories, a better understanding of velocity, and estimates.

Weyrauch [19] describes the agile adoption at Medtronic. One of the barriers the team faced in their agile adoption was the perception that agile projects do not need to plan. The team worked to correct that mistaken impression and the result was the new perception that agile projects are all about planning and re-planning constantly, "the exact opposite of the original worry" [19].

Honious [20] describes the path of a product Reed Elsevier was developing. The team was working on a release plan that would result in the product being demonstrated at a tradeshow. Reality, however, intruded on those initial plans, and it soon became clear that the team would not be able to make the deadline with the current scope and constraints. The team worked with the senior stakeholders to re-plan. Honious emphasizes that it was unacceptable to just "drop functionality" as there was a minimal feature set required for the tradeshow, so the team developed solutions with this constraint in mind. The feature set was already the minimal acceptable for the business case, so the team's re-planned solution deferred features that were not required for the tradeshow users (but would be required for the full release), and they also added another pair to the development team.

Experience shows that initial plans are often optimistic and customers will need to regularly re-plan. Re-Calibration allows stakeholders to make changes to the plan regularly as they learn more about the project, thus supporting Real Customer Involvement. Re-Calibration contributes to creating a Whole Team as they move away from a "blame" culture towards a proactive and regular re-planning event. Finally, Re-Calibration also contributes to Energized Work as it ensures they re-plan the work for the whole team, including the customer.

11. Conclusion

This paper has outlined the customer-focused practices that emerge from our qualitative study of XP teams, also identifying the inherent interwoven relationships between the practices, and how they strongly contribute to Real Customer Involvement, Whole Team and Energized Work.

The emergent practices primarily support Real Customer Involvement by preparing the business representatives for their role (Customer Boot Camp), and providing opportunities for the business representatives to contribute towards the creation and refinement of what to build (Big Picture Up-Front, Roadshow and Re-calibration). The emergent practices primarily support Whole Team by providing opportunities for the programmers to develop empathy for the customer team (Customer's Apprentice) and the end-user (Programmer On-Site). Finally, the emergent practices primarily contribute to Energized Work by reducing the intensity of the process (Pair Customering and Programmer Holiday). As with all of the XP practices, the emergent customer practices are not specific solutions, but rather focus on describing how to support teams to work together more effectively and how to ensure they "build the right thing". We hope this qualitative work has identified good ideas to help other teams, and frame opportunities for further research.

12. References

[1] M. Fowler. (2002), XP Customer Quotes, WikiWikiWeb, http://martinfowler.com/articles/agileStory.html.

[2] K. Beck. (2004), eXtreme Programming Explained: Embrace Change. Second Edition, Addison-Wesley.

[3] K. Beck. (2000), eXtreme Programming Explained: Embrace Change, Addison-Wesley.

[4] B. Fitzgerald. (2000), Systems Development Methodologies: The Problem of Tenses. Information Technology and People 13 174 - 185.

[5] J. Nandhakumar, and D.E. Avison. (1999), The Fiction of Methodological Development: A Field Study of Information Systems Development. Information Technology and People 12 1 - 28.

[6] B.G. Glaser, and A.L. Strauss. (1967), The Discovery of Grounded Theory: Strategies for Qualitative Research., Chicago: Aldine.

[7] A. Martin. (2009), Exploring the Role of Customers in Extreme Programming Projects. PhD Thesis, Victoria University of Wellington, New Zealand. http://researcharchive.vuw.ac.nz/handle/10063/877.

[8] A. Martin, R. Biddle, and J. Noble. (2009), The XP Customer Team: A Grounded Theory. *in* "Agile 2009", IEEE Computer Society, Chicago.

[9] P. Merholz, T. Wilkens, B. Schauer, and D. Verba. (2008), Subject To Change: Creating Great Products and Services for an Uncertain World, O'Reilly.

[10] P. Hodgetts. (2004), Refactoring the Development Process: Experiences with the Incremental Adoption of Agile Processes. *in* "Agile Development Conference" (T. Little, S. Alpert, and A. Pols, Eds.), IEEE Computer Society, Salt Lake City, Utah, United States.

[11] A. Takats, and N. Brewer. (2005), Improving Communication between Customers and Developers. *in* "Agile 2005" (M.L. Manns, and W. Wake, Eds.), IEEE Computer Society, Denver, Colarado, United States.

[12] H. Beyer, K. Holtzblatt, and L. Baker. (2004), An Agile Customer-Centered Method: Rapid Contextual Design. *in* "XP Agile Universe", Calgary, Alberta, Canada.

[13] D. Broschinsky, and L. Baker. (2008), Using Persona with XP at LANDesk Software, an Avocent Company. *in* "Agile 2008" (G. Melnik, P. Kruchten, and M. Poppendieck, Eds.), IEEE Computer Society, Toronto, Canada.

[14] T. Mackinnon. (2003), XP - Call in the Social Workers. *in* "Fourth International Conference on eXtreme Programming and Agile Processes in Software Engineering." (G. Succi, Ed., Springer-Verlag., Genoa, Italy.

[15] K. Schwaber, and M. Beedle. (2001), Agile Software Development with Scrum, Prentice-Hall.

[16] M. Ganis, D. Leip, F. Grossman, and J. Bergin. (2005), Introducing Agile Development (XP) into a Corporate Webmaster Environment - An Experience Report. *in* "Agile 2005" (M.L. Manns, and W. Wake, Eds.), IEEE Computer Society, Denver, Colorado, United States.

[17] J. Rasmusson. (2006), Agile Project Initiation Techniques - The Inception Deck and Boot Camp. *in* "Agile 2006" (J. Chao, M. Cohn, F. Maurer, H. Sharp, and J. Shore, Eds.), IEEE Computer Society, Minneapolis, United States.

[18] A.M. Fuqua, and J.M. Hammer. (2003), Embracing Change: An XP Experience Report. *in* "Fourth Internal Conference on Extreme Programming and Agile Processes in Software Engineering" (M. Marchesi, and G. Succi, Eds.), Springer-Verlag, Genoa, Italy.

[19] K. Weyrauch. (2006), What Are We Arguing About? A Framework for Defining Agile in our Organization. *in* "Agile 2006" (J. Chao, M. Cohn, F. Maurer, H. Sharp, and J. Shore, Eds.), IEEE Computer Society, Minneapolis, United States.

[20] J. Honious, and J. Clark. (2006), Something to Believe In. *in* "Agile 2006" (J. Chao, M. Cohn, F. Maurer, H. Sharp, and J. Shore, Eds.), IEEE Computer Society, Minneapolis, United States.